EE604_python_basics

September 2, 2023

[6]:
```
# IITK EE604A Image Processing
#
# Author: Dr. Tushar Sandhan
#
# In response to a request from a few students, I am creating this basic␣
 ↪hands-on tutorial to get you started.
# Try to learn additional Python functions throughout the course's assignments;␣
 ↪do not limit yourself to these functions alone.
```

[7]:
```
# Image reading libraries
import cv2
import matplotlib.image as mpimg
from PIL import Image

# each one of above has of its own format for storing image variable after␣
 ↪reading it from a file
```

[8]:
```
# Once you read images then you can convert them to numpy arrays / matrices
# (or can direct get numpy arrays / matrices from one of the above image readers)
import numpy as np
```

[9]:
```
# intermittant plotting library for graphs and can display images also
import matplotlib.pyplot as plt
```

[10]:
```
# image file in the array using PIL package (you can check other image readers␣
 ↪as well)
input_img = np.array(Image.open('sample_input_image.jpg'))  # give full path to␣
 ↪input image
```

[11]:
```
print(input_img.shape)
```

```
(92, 156, 3)
```

[12]:
```
# If we want to use same PIL package for writing image matrix as a image file
# then first we have to convert numpy array to unsigned int (byte) and to PIL␣
 ↪format
pilimg = Image.fromarray(np.uint8(input_img))
pilimg.save('resaved_input_image.jpg')
```

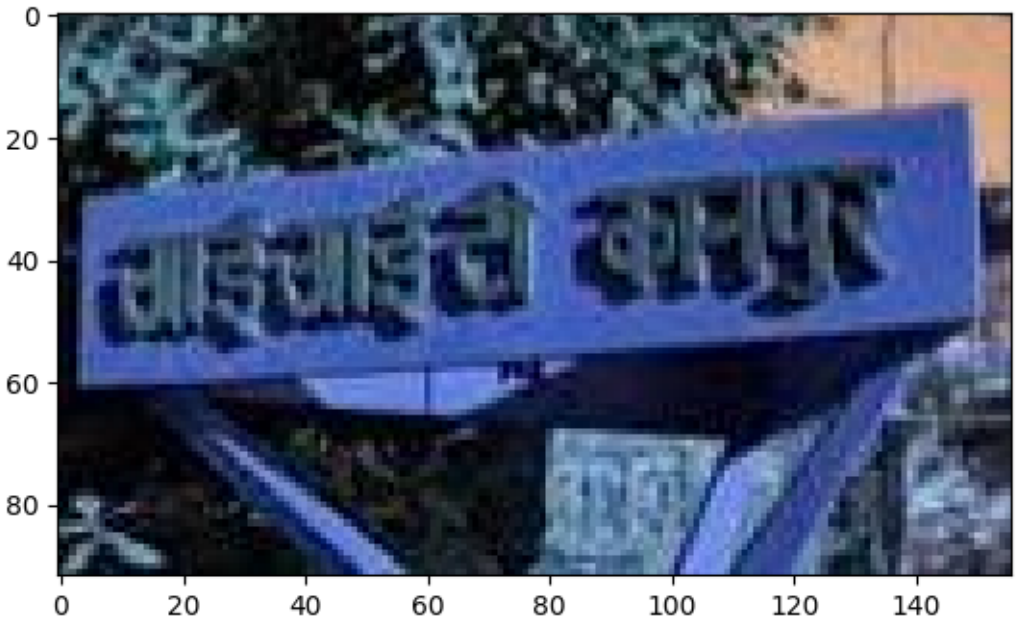
```
# check your saved and original image in your PC or
# download from google colab (if you are running these programs on google colab)
 ↪and check it on your PC
```

[13]:
```
img_orig = cv2.imread("resaved_input_image.jpg")
# Note: OpenCV loads images in 'BGR' format.
# img_orig[:, :, 0] -- corresponds to blue part of the image
# img_orig[:, :, 1] -- corresponds to green part of the image
# img_orig[:, :, 2] -- corresponds to red part of the image
print(img_orig.shape)
```

```
(92, 156, 3)
```

[14]:
```
plt.imshow(img_orig)  #This is what happens when one reads BaaziGaR but another
 ↪considers it as RanGdeBasanti
```

[14]:
```
<matplotlib.image.AxesImage at 0x7cb27cdeb760>
```



[15]:
```
img = img_orig[:, :, [2, 1, 0]] # change to rgb
```

[16]:
```
plt.imshow(img)
plt.axis("off") # disable axis
plt.show()
```

[17]:
```
img = cv2.imread('sample_input_image.jpg', 0) # loads directly greyscale version
 ↪of the image
print(img.shape) # no color channel

plt.imshow(img, cmap='gray') # you need to inform pyplot that you want a
 ↪grayscaled image
plt.axis("off")
plt.show()
```

```
(92, 156)
```

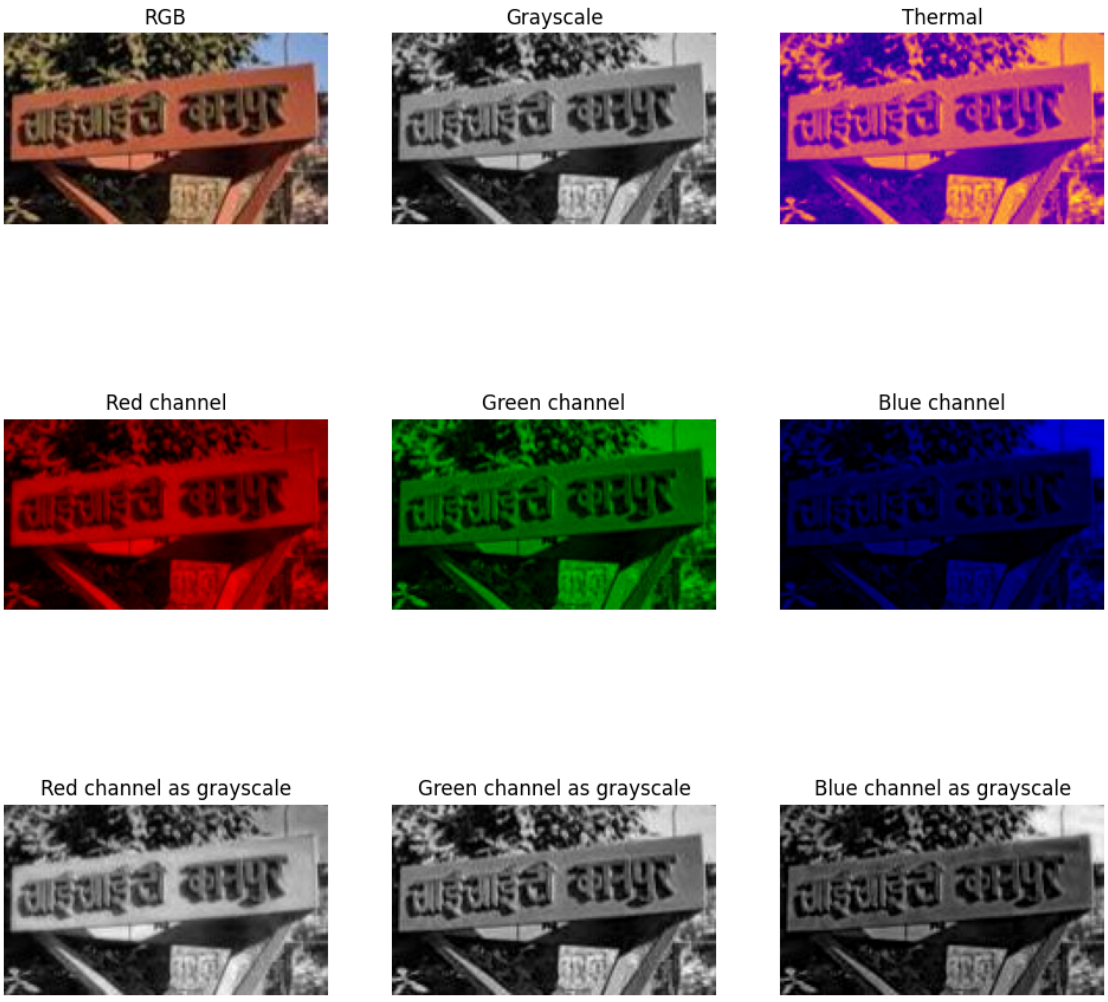
[18]:
```
# lets visualize IITK via thermal display
plt.imshow(img, cmap="plasma")
plt.axis("off")
plt.colorbar() # display the colorbar
plt.show()
#IITK is always hot (academically as well as environmentally)
```

[19]:
```
#cv2 can also show something
from google.colab.patches import cv2_imshow  #google colab only uses this patch,␣
↪otherwise you won't need it in local PC
cv2_imshow(img)
```

[20]:
```
cv2_imshow(img_orig)
# No need to convert to RGB, google colab's patch handles cv2's insanity␣
 ↪internally
```

[21]:
```
#Display multiple images in the same figure
img_orig = cv2.imread("sample_input_image.jpg")

# convert to greyscale
img_gray = cv2.cvtColor(img_orig, cv2.COLOR_BGR2GRAY)
# convert to rgb
img_rgb = cv2.cvtColor(img_orig, cv2.COLOR_BGR2RGB)

img_blue_channel = img_orig[:, :, 0] # blue channel
img_green_channel = img_orig[:, :, 1] # green channel
img_red_channel = img_orig[:, :, 2] # red channel

#we copy data to avoid modifications in original image numpy array as numpy␣
 ↪arranges are accessed via referencing
img_blue, img_green, img_red = np.copy(img_rgb), np.copy(img_rgb), np.
 ↪copy(img_rgb)
img_blue[:, :, [0, 1]] = 0 # set values of rest of the channels to be zero so␣
 ↪that we see color corresponding to single channel
img_green[:, :, [0, 2]] = 0 # set values of rest of the channels to be zero so␣
 ↪that we see color corresponding to single channel
img_red[:, :, [1, 2]] = 0 # set values of rest of the channels to be zero so␣
 ↪that we see color corresponding to single channel

plt.figure(figsize=(12, 12)) #Initiate figure

plt.subplot(3, 3, 1) #having 3x3grids (on each grid we will display one image)
plt.imshow(img_rgb)
plt.axis("off")
plt.title("RGB")

plt.subplot(3, 3, 2)
```

```
plt.imshow(img_gray, cmap="gray")
plt.axis("off")
plt.title("Grayscale")

plt.subplot(3, 3, 3)
plt.imshow(img_gray, cmap="plasma")
plt.axis("off")
plt.title("Thermal")

plt.subplot(3, 3, 4)
plt.imshow(img_red)
plt.axis("off")
plt.title("Red channel")

plt.subplot(3, 3, 5)
plt.imshow(img_green)
plt.axis("off")
plt.title("Green channel")

plt.subplot(3, 3, 6)
plt.imshow(img_blue)
plt.axis("off")
plt.title("Blue channel")

plt.subplot(3, 3, 7)
plt.imshow(img_red_channel, cmap="gray")
plt.axis("off")
plt.title("Red channel as grayscale")

plt.subplot(3, 3, 8)
plt.imshow(img_green_channel, cmap="gray")
plt.axis("off")
plt.title("Green channel as grayscale")

plt.subplot(3, 3, 9)
plt.imshow(img_blue_channel, cmap="gray")
plt.axis("off")
plt.title("Blue channel as grayscale")

plt.show()
```



[21]:

Numpy resources: https://numpy.org/devdocs/user/absolute_beginners.html

[40]:
```
from PIL import Image, ImageFilter


img = Image.open('sample_input_image.jpg')
img = img.convert("L")
ycbcr = img.convert("YCbCr")

plt.imshow(img, cmap="gray")
plt.axis("off")
plt.show()
```

[ ]:

[38]:
```
filtered = img.filter( ImageFilter.Kernel( (3,3), (1/9,1/9,1/9, 1/9, 1/9, 1/9, 1/
↪9, 1/9, 1/9), 1, 0) )

plt.imshow(filtered, cmap="gray")
plt.axis("off")
plt.show()

filtered.save("Edge_sample_input_image.jpg")
```

[37]:
```
from skimage.io import imread, imshow, imsave
from skimage.color import rgb2gray
from scipy.signal import convolve2d

identity = np.array([ [1/9,1/9,1/9],
                      [1/9, 1/9,1/9],
                      [1/9,1/9,1/9]])

conv_im1 = convolve2d(img, identity)


plt.imshow(conv_im1, cmap="gray")
plt.axis("off")
plt.show()
```

[ ]:
```
import argparse

parser = argparse.ArgumentParser(description='Parser for command line arguments')

parser.add_argument('pos_arg', type=int, help='Input integer for image␣
 ↪processing program')

args = parser.parse_args()
```

[ ]:
```
!python filtering_program.py 5
```